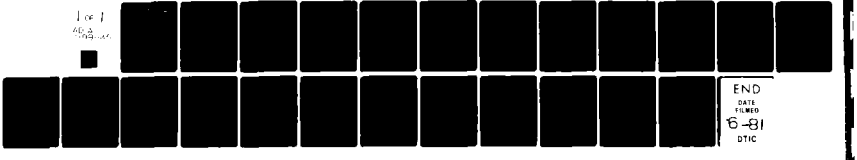
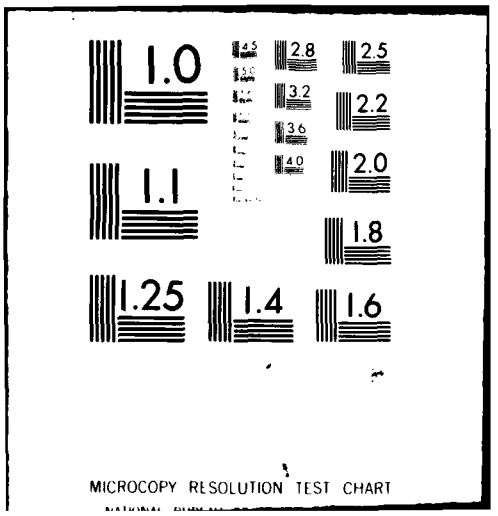
AD-A098 835 STANFORD UNIV CA DEPT OF COMPUTER SCIENCE F/G 12/1
TWO LINEAR-TIME ALGORITHMS FOR FIVE-COLORING A PLANAR GRAPH,(U)
NOV 80 D W MATULA, Y SHILOACH, R E TARJAN N00014-76-C-0330
UNCLASSIFIED STAN-CS-80-830 NL



END
DATE FILMED
6-81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

Stanford Department of Computer Science
Report No. STAN-CS-80-830

(11) Nov[redacted]80

(12) 24

(6)

TWO LINEAR-TIME ALGORITHMS FOR FIVE-COLORING A PLANAR GRAPH

by (W.)

(10) David Matula
Yossi Shiloach
Robert Tarjan

(E.)

(15) N00014-76-C-0[illegible]
N[illegible]41-7[illegible]

Research sponsored by

National Science Foundation
and
Office of Naval Research

DEPARTMENT OF COMPUTER SCIENCE
Stanford University

Two Linear-Time Algorithms for Five-Coloring a Planar Graph

David W. Matula* †
Computer Science Department
Stanford University
Stanford, California 94305

Yossi Shiloach
Israel Scientific Center
IBM Israel Ltd.
Haifa, Israel

Robert E. Tarjan**
Computer Science Department
Stanford University
Stanford, California 94305

August 1980

Abstract

A "sequential processing" algorithm using bicolor interchange that five-colors an n vertex planar graph in $O(n^2)$ time was given by Matula, Marble, and Isaacson [MMI 72]. Lipton and Miller used a "batch processing" algorithm with bicolor interchange for the same problem and achieved an improved $O(n \log n)$ time bound [LM 78]. In this paper we use graph contraction arguments instead of bicolor interchange and improve both the sequential processing and batch processing methods to obtain five-coloring algorithms that operate in $O(n)$ time.

Keywords: algorithm, batch processing, coloring, contraction, data structures, inductive proof, linear time, planar graph, recursion, sequential processing.

* Research partially supported by National Science Foundation grant MCS-77-23738 and by Office of Naval Research contract N00014-76-C-0330. Reproduction in whole or in part is permitted for any purpose of the United States government.

** Research partially supported by National Science Foundation grant MCS-782-6858 and by Office of Naval Research contract N00014-76-C-0688.

† On sabbatical leave from the Department of Computer Science/Engineering, Southern Methodist University, Dallas, Texas 75275.

1. Introduction and Summary.

We present two different linear-time algorithms for coloring the vertices of a planar graph with at most five colors. Our methods utilize a recursive graph contraction process to order the vertices followed by a color assignment pass through the ordered vertices (no assigned color values are ever changed).

Several variations of a "sequential processing" coloring algorithm were previously given in the literature [MMI 72]. These algorithms 6-color any planar graph in $O(n)$ time and 5-color such a graph in $O(n^2)$ time. Lipton and Miller [LM 78] have employed a "batch processing" approach to obtain an improved planar graph 5-coloring algorithm operating in $O(n \log n)$ time (see also [S 79]). In this paper we improve both the sequential processing and batch processing algorithms by using graph contraction to obtain linear-time behavior in each case.

The sequential processing and batch processing algorithms are paradigms representing different philosophies in algorithm design and it is worthwhile to contrast how linear time is achieved in each case. Both paradigms employ recursion to reduce the problem (planar graph 5-coloring) on an n vertex graph to the same problem on an $n' < n$ vertex graph. For each successive reduction the sequential processing paradigm utilizes time bounded by $c(n - n')$ for some fixed constant c. An overall linear time bound is thus obtained even though the number of reductions may itself be of linear order. Each reduction in the batch processing paradigm can use time up to $c_1 n$ for some fixed constant c_1, but each such reduction involves a sufficient proportion of the vertices to obtain $n' < c_2 n$ for some fixed constant $c_2 < 1$. An overall linear time bound is obtained in this case since the total reduction time is less than $\sum_{i=0}^{\infty} c_1 c_2^i n = c_1 n/(1 - c_2)$.

In the next section we develop some preliminary graph coloring theorems and define contraction. We then give a concise proof of planar graph 5-colorability utilizing the contraction argument. In Section 3 we present a linear-time planar graph 5-coloring algorithm of the sequential processing type which is motivated by the preceeding contraction-based 5-colorability proof. A batch processing type linear-time planar graph 5-coloring algorithm is developed in Section 4. The final section surveys some issues arising from our algorithms.

2. Graph Coloring Theorems and Graph Contraction.

The algorithms we develop, as well as those algorithms previously cited [MMI

72, LM 78], can be closely associated with specific inductive proofs of planar graph 5-colorability. The realization of an efficient recursive algorithm from an inductive proof requires careful design and analysis of appropriate data structures. An elementary example of this process is given by the following linear-time planar 6-coloring algorithm [MMI 72], which will be needed later.

Lemma 1. *Every planar graph can be colored with at most six colors.*

Inductive Proof. Every graph on $n \leq 6$ vertices can be 6-colored. Assume every graph on at most n vertices can be 6-colored for a given $n \geq 6$, and let the planar graph G have $n+1$ vertices. From Euler's Theorem [H 68, p. 103] it follows that some vertex v of G has degree at most five. By hypothesis $G - v$ can be six-colored and v may then be assigned one of the six colors not occuring on its at most five adjacent vertices. The lemma follows by induction. □

ALGORITHM 6-COLOR.

Given an n vertex planar graph G in adjacency list form, this algorithm determines a 6-coloring of G.

Step 1. [Establish degree lists.] For each j where $0 \leq j \leq n-1$, form a doubly linked list of all vertices of G of degree j.

Step 2. [Label vertices smallest degree last.] For $i = n, n-1, n-1, \ldots, 1$ designate the first vertex of the non-vacuous j degree list of smallest j as vertex v_i. Delete v_i from the j degree list. For each vertex v' that was adjacent to v_i in G and remains in some degree list, say j', delete v' from the j' degree list and insert v' in the $j'-1$ degree list.

Step 3. [Color vertices.] For $i = 1, 2, \ldots, n$, assign vertex v_i the smallest color value (which must be some integer between one and six) not occuring on the vertices adjacent to v_i that have already been colored.

Claim. *Algorithm 6-COLOR determines a 6-coloring of any n vertex planar graph and can be implemented to run in $O(n)$ time.*

Proof. In Algorithm 6-COLOR each vertex v_i, for $1 \leq i \leq n$, is chosen so that it has minimum degree in the planar graph $G - \{v_{i+1}, v_{i+2}, \ldots, v_n\}$. Hence v_i is adjacent to at most five previously colored vertices when its color is assigned

in Step 3, so Algorithm 6-COLOR yields a 6-coloring of any planar graph. To show that Algorithm 6-COLOR can be implemented with a linear time bound, first note that the adjacency list data structure has length of order the number of edges of G which is $O(n)$ since the number of edges is at most $3n - 3$ for any planar n vertex graph. Establishment of the degree structure in Step 1 requires $O(n)$ time. For the processing of vertex v_i in Steps 2 and 3 let $degree(v_i)$ refer to the degree of v_i in the original n vertex graph G, so $degree(v_i)$ is the length of the adjacency list for v_i. In Step 2 the determination of the vertex labelled v_i requires $O(j) = O(degree(v_i))$ time. Deletion of v_i and the deletion and reinsertion of the at most $degree(v_i)$ remaining vertices adjacent to v_i can be accomplished in $O(degree(v_i))$ time (by traversing the adjacency list of v_i). Step 2 thus requires $O(\sum_{i=1}^{n} degree(v_i)) = O(n)$ time since $\sum_{i=1}^{n} degree(v_i)$ equals twice the number of edges of G. Scanning the adjacency list of v_i to determine the color of v_i in Step 3 requires time $O(degree\ v_i)$, so Step 3 can also be realized in $O(\sum_{i=1}^{n} degree(v_i)) = O(n)$ time. □

In Algorithm 6-COLOR the output of Step 2 is an ordering of the vertices $v_1, v_2, \ldots, v_n$ of the planar graph G such that v_i is adjacent to at most five preceeding vertices v_j, $j < i$. Step 3 then computes a 6-coloring of G from this ordering in $O(n)$ time. There is a generalization of this result that will be used for the color assignment phase of our subsequent algorithms. An *independent* set I of vertices of G has no pair of vertices of I adjacent. Two disjoint independent sets I, J of G are termed *adjacent* whenever some vertex in I is adjacent to some vertex in J.

Lemma 2. *Let G be a graph with n vertices and ℓ edges. Let $I_1, I_2, \ldots, I_p$ be a partition of the vertices of G into independent sets where each I_k for $1 \leq k \leq p$ is adjacent to at most j sets $I_{k'}$ with $k' < k$. Then G may be $(j+1)$-colored by an algorithm requiring $O(n + \ell)$ time.*

Proof. Let G be given in adjacency list form. Color the vertices of G by the following procedure. Color all vertces of I_1 with color value 1. For $k = 2, 3, \ldots, p$, traverse the adjacency list of all vertices of I_k to find which color values of the set $\{1, 2, \ldots, j+1\}$ have already been assigned to vertices adjacent to vertices of I_k. There can be at most j such colors on adjacent vertices since I_k is adjacent to at

most j sets $I_{k'}$. Thus G is colored in $j+1$ colors. The coloring process described requires traversing the adjacency list of each vertex once, which requires $O(n+\ell)$ time, along with other activities taking $O(n)$ time. Thus the coloring procedure is realizable by an algorithm requiring $O(n+\ell)$ time. □

Both of our subsequent algorithms in their contraction phase will determine from a planar graph G a partition of the vertices into independent sets $I_1, I_2, \ldots, I_p$, where for any $1 \leq k \leq p$, the set I_k is adjacent to at most four sets $I_{k'}$ with $k' < k$. The linear-time color-assignment phase of each algorithm is then obtained from Lemma 2.

The standard proof of 5-colorability of planar graphs utilized in graph theory texts [B 62, p. 213; H 68, p. 130; BC 71, p. 223; BM 76, p. 156; B 79, p. 95] is due to Heawood [H 90] published in 1890. The argument is inductive and similar to the preceeding proof of 6-colorability with the following extension. When a vertex v of degree five is to be added to the 5-colored graph $G-v$, an appropriate color interchange [interchanging colors i and j on a maximal connected subgraph of vertices colored i or j] is shown to exist if needed to guarantee that v has only four distinct colors occuring on the five adjacent vertices. An algorithm incorporating this color interchange step via sequential processing was given by Matula, Marble and Isaacson [MMI 72] yielding a planar graph 5-coloring algorithm requiring $O(n^2)$ time. Lipton and Miller [LM 78] applied a batch processing approach to the color interchange step to achieve an $O(n \log n)$ 5-coloring algorithm. We now define graph contraction and present a concise proof of planar graph 5-colorability based on contraction. This proof motivates our linear time 5-coloring algorithms.

An *elementary contraction* of a graph G is obtained by identifying two adjacent vertices u, v and the edge uv as a new vertex w. More specifically, we perform such a contraction by removing u and adding a new vertex w adjacent to those vertices previously adjacent to u or v. A graph H is a contraction of G if H can be obtained from G by a sequence of elementary contractions. Note that a contraction of a planar graph is also a planar graph [e.g. see LT 79].

Theorem 3. *Every planar graph is 5-colorable.*

Proof. Every graph on five or less vertices can be colored in at most five colors. Assume every planar graph on n vertices can be five-colored for a given

$n \geq 5$, and let the planar graph G have $n+1$ vertices. It follows from Euler's theorem that some vertex v of G has degree at most five. If $deg(v) \leq 4$, by hypothesis $G - v$ can be 5-colored and then v may be assigned one of the five colors not occuring on the at most four vertices adjacent to v. If $deg(v) = 5$, then there are two non-adjacent vertices u, w among the five vertices adjacent to v, since five vertices cannot all be pairwise adjacent in a planar graph. The contraction H of G obtained by identifying the three vertices u, v, w is then 5-colorable by hypothesis. Assign every vertex in G other than $\{u, v, w\}$ the same color as in this 5-coloring of H and assign both u and w the color of the vertex in H created by the contraction of u, v, and w. This provides a 5-coloring of $G - v$ in which at most four distinct colors appear on the five vertices adjacent to v, so G may be 5-colored. By induction the theorem holds. □

3. The Sequential Contraction Algorithm.

The proof of 5-colorability in Theorem 3 differs from the proof of 6-colorability in Lemma 1 only when the planar graph encountered in the induction has minimum degree five. In this case contraction is used rather than vertex deletion. We now investigate whether the contraction process can be implemented in a sufficiently efficient manner so that we can improve Algorithm 6-COLOR to achieve a linear time 5-coloring algorithm. Thus we seek to determine whether the following two steps can be implemented in time bounded by a constant whenever the algorithm through recursion encounters a planar graph of minimum degree five.

Contraction Process

Step 1. Find a vertex v of degree five and two non-adjacent vertices u, w among the five vertices adjacent to v;

Step 2. Update the appropriate data structures to correspond to the structure of the graph resulting from the contraction.

It is perhaps surprising to realize that implementation of Step 1 in constant time is a non-trivial problem. Certainly determining a vertex v of degree five along with its remaining adjacent vertices u_1, u_2, u_3, u_4, u_5 in the reduced graph can be accomplished by using degree lists as in Algorithm 6-COLOR, at a cost of

constant time per vertex. The problem arises in determining a non-adjacent pair among the vertices u_1, u_2, u_3, u_4, u_5. This would require only constant time if the adjacency matrix were available, but constructing the adjacency matrix for an n vertex graph requires $\Omega(n^2)$ time and is therefore precluded. Observe that if only the adjacency list data structure of a graph is available, then determination of whether or not a pair u_i, u_j of vertices are adjacent requires time proportional to $\min\{degree(u_i), degree(u_j)\}$ since the adjacency list of either vertex u_i or u_j must be scanned to determine the presence or absence of the other vertex.

The preceding observation dictates that we must be concerned with the vertex degrees of the vertices adjacent to a vertex of degree five in implementing Step 1 of the contraction process. The graph G_n of Figure 1 suggests a possible complication.

[Figure 1]

The class of graphs G_n, for $n = 14, 16, 17, \ldots$, illustrates that the sum of the vertex degrees of the vertices adjacent to any given vertex in a planar graph may be arbitrarily high. In order to design our algorithm we shall need an additional result on the structure of planar graphs, which is stated in the following theorem.

Theorem 4. *Every planar graph has either:*

(i) *a vertex of degree at most four, or*

(ii) *a vertex of degree five adjacent to at least four vertices each having degree at most eleven.*

We shall defer a proof of Theorem 4 until the end of this section, and first give a planar graph 5-coloring algorithm based on the result of Theorem 4.

It should be noted that the following algorithm operates on a planar graph in a purely combinatorial manner without reference to any particular embedding.

ALGORITHM SC [Sequential Contraction 5-Coloring]

Given an n vertex planar graph G in adjacency list form, this algorithm determines a 5-coloring of G. In the following description G refers to the original planar graph; and G^* refers to a graph that initially has the same structure as G, but which is reduced by deletions and contractions until it is the null graph.

At all times each vertex of G^* corresponds to an independent set of vertices of G.

Step 1. [Establish adjacency structure and degree lists for G^*.] Create an adjacency list data structure for the graph G^* which is initially the same graph as G. For each $0 \leq j \leq n-1$ with $j \neq 5$, form a list of all vertices of G^* of degree j (the j-degree lists). Form two distinct lists for vertices of degree five depending on whether or not the degree five vertex has at least four adjacent vertices of degree at most eleven (respectively the $5^{[11]}$-degree and 5-degree lists).

Step 2. [Delete the minimum degree vertex in G^* if degree is at most four.] If the j-degree lists for $0 \leq j \leq 4$ are all vacuous, go to Step 3. Otherwise, determine v^* to be the first vertex in the non-vacuous j-degree list of smallest j. Delete v^* from G^* and add v^* to the stack of vertices deleted. Update the adjacency list data structure and degree lists for G^* to reflect the deletion of v^*. Repeat Step 2.

Step 3. [Contraction process for G^* of minimum degree five.] If G^* is not null determine v^* to be the first vertex in the $5^{[11]}$-degree list. Among the four vertices adjacent to v^* in G^* having degree at most eleven, determine a non-adjacent pair u^*, w^*. Delete v^* from G^* and add v^* to the stack of vertices deleted. Update the adjacency list data structure and degree lists to correspond first to the deletion of v^* from G^*, and then to the contraction of u^* and w^* into a new vertex y^* of G^*, where y^* represents the union of the independent sets of G represented by u^* and w^*. If G^* is still not null, go to Step 2.

Step 4. [Color vertices of G.] While the stack of vertices deleted from G^* is not empty, remove the top vertex v^* from the stack. Traverse the independent set $\{v_1, v_2, \ldots, v_k\}$ of vertices of G represented by v^* and color all these vertices with the same smallest positive integer value (which must be between one and five) that has not previously been assigned as the color of any vertex of G adjacent to any vertex of the set $\{v_1, v_2, \ldots, v_k\}$.

Theorem 5. *Algorithm SC determines a 5-coloring of any n vertex planar graph.*

Proof. The following two statements are easily verified by induction on the number of steps executed by the algorithm. As the algorithm proceeds, each vertex of G^* corresponds to an independent set I_{v^*} of vertices in G, such that two vertices v^* and w^* of G^* are adjacent if and only if I_{v^*} and I_{w^*} are adjacent in G. G^* remains planar throughout the algorithm. Let v^* be a vertex deleted from G^* during an execution of either Step 2 or Step 3. Then I_{v^*} is adjacent in G to at most four independent sets corresponding to vertices in the new graph G^* resulting from the execution of Step 2 or Step 3. (Note that Step 3 not only deletes v^* but contracts two other vertices.) By Theorem 4, the $5^{[11]}$-degree list is never empty when the algorithm reaches Step 3. If v^* is a vertex on the $5^{[11]}$-degree list, at least two of the four neighbors of v^* with degree eleven or less are non-adjacent (otherwise G^* would contain a 5-clique and would be non-planar). It follows that the algorithm can never get stuck in Step 3, and it repeats Steps 2 and 3 until G^* is null. If $v_1^*, v_2^*, \ldots, v_k^*$ are the successive vertices deleted from G^* by Steps 2 and 3, $I_{v_1^*}, I_{v_2^*}, \ldots, I_{v_k^*}$ is a partition of G^* into independent sets such that any $I_{v_i^*}$ is adjacent to at most four sets $I_{v_j^*}$ with $j > i$. By Lemma 2, Step 5 successfully 5-colors G. □

We now specify the data structures to be used in a linear-time implementation of Algorithm SC. We assume that G is represented initially by a singly-linked adjcency list for each vertex. The adjacency list data structure created for G^* is doubly linked and has *crosspointers* for each edge, i.e., vertex u^* in the list of vertex v^* has a pointer to vertex v^* in the list of vertex u^* and vice-versa. A vertex v^* of G^* points to a circular list consisting of the vertices in the corresponding independent set of G. Note that constructing the union of two disjoint circular lists requires only O(1) time. Initially each vertex v of $G^* = G$ points to a list containing the single element v. The degree lists for G^* are doubly linked. The stack of deleted vertices is initially empty.

Theorem 6. *Algorithm SC can be implemented to run in* $O(n)$ *time.*

Proof. Given as input the adjacency list of an n vertex planar graph, the data structures described prior to the statement of Theorem 6 can be constructed in Step 1 of Algorithm SC in $O(n)$ time. We shall now show that the execution of either Step 2 or Step 3 resulting in the deletion of a vertex v^* from G^* can be completed in time bounded by a constant. We must consider two cases.

Case 1. G^* has minimum degree at most four. Then the vertex v^* to be deleted is found in Step 2 in O(1) time by extraction from the non-vacuous j-degree list of smallest j, where $0 \leq j \leq 4$. By traversing at most four vertices of the adjacency list of v^* and using the crosspointers, the adjacency structure for G^* can be updated to reflect deletion of v^* in O(1) time. During the same traversal of the adjacency list of v^*, each vertex w^* adjacent to v^* can be deleted from its j-degree list and inserted in the $(j-1)$-degree list. There are two special cases:

(i) $j = 6$ Then the degrees of the five vertices other than v^* adjacent to w^* should be noted in order to insert w^* in the appropriate $5^{[11]}$- or 5-degree list;

(ii) $j = 12$ Then the eleven vertices other than v^* adjacent to w^* should have their degrees checked and if any, say u^*, is in the 5-degree list, its five adjacent vertices should have their degrees observed to see if u^* should be moved to the $5^{[11]}$-degree list as w^* is reduced from degree twelve to degree eleven.

Note that all such updating of the degree structure requires only O(1) time.

Case 2. G^* has minimum degree five. Then v^* is extracted from the $5^{[11]}$-degree list in Step 3. The adjacency lists of the four vertices of degree at most eleven adjacent to v^* are traversed in O(1) time to find a non-adjacent pair u^*, w^*. The updating of the data structures corresponding to the deletion of v^* from G^* in Step 3 can be implemented in O(1) time by the same arguments used in Case 1. It remains to consider implementation of the contraction.

The circular list representing I_{y^*} is the disjoint union of the lists representing I_{u^*} and I_{w^*} and can be constructed in O(1) time. To update the adjacency structure for G^* we initialize the adjacency list of y^* to be the list of u^*, and we traverse this list to change the crosspointers of other entries that previously pointed to u^* so that they now point to y^*. We then take each vertex x^* of the adjacency list of w^* and search the adjacency list of y^* to see if x^* is already there. If not, we add x^* to the adjacency list of y^* and correct the crosspointers in x^*'s list to point to y^* rather than w^*. If x^* is already in the list of y^*, corresponding to the former edges x^*u^* and x^*w^* becoming a single edge x^*y^* after contraction,

then the crosspointer of w^*x^* is used to delete what would otherwise become a second reference to y^* in x^*'s list. Since the lengths of the adjacency lists of u^* and w^* are both at most eleven, this updating of the adjacency structure for G^* requires at most O(1) time.

As we update the adjacency structure for G^* we also update the degree lists for G^* to reflect the contraction. Vertices u^* and w^* are deleted from their degree lists and y^* is inserted into its appropriate degree list depending on its resulting degree, which is somewhere between four and twenty. Any vertex x^* now adjacent to y^* need be moved in the degree lists only if either x^* was previously adjacent to both u^* and w^*, in which case it is removed from its j degree list and inserted in the $j-1$ degree list (handling the appropriate special cases $j = 6$ and $j = 12$ as in the discussion of Case 1), or if x^* has degree five, in which case the degrees of the vertices adjacent to x^* are observed to assure placement of x^* in the appropriate $5^{[11]}$- or 5-degree list. The total number of vertices encountered by all such references in the contraction phase whose position in the degree lists may have to be changed is at most 132. Figure 2 indicates the relevant vertices.

[Figure 2]

Thus the updating of data structures for each execution of Step 3 requires O(1) time.

By Theorem 4, Cases 1 and 2 are exhaustive. Since at most n deletions from G^* can occur in either Step 2 or Step 3, this implementation has a total time of $O(n)$ for Steps 2 and 3. Finally, the coloring in Step 4 of Algorithm SC can be implemented by Lemma 2 in at most time proportional to the number of edges of G, which is $O(n)$. Thus Algorithm SC provides a 5-coloring in linear time. □

The linear time bound realized in our implementation of Algorithm SC depends critically on the result in Theorem 4 that the vertices u^*, w^* selected to be merged by contraction in Step 3 have bounded degrees. We now prove Theorem 4.

Proof of Theorem 4. Since any planar graph G has minimum degree at most five, it is sufficient to show that if G has minimum degree exactly five, then some vertex of degree five in G is adjacent to at least four vertices of degree at most eleven. Without loss of generality we may assume that G is maximal planar.

Let A be the set of vertices of G of degree five with $n_A = |A| \geq 1$, and let B be the set of vertices of G of degree twelve or greater, with $n_B = |B|$. The average degree of any planar graph is strictly less than six, so

$$n_A > 6n_B. \tag{1}$$

Assume each vertex of A is adjacent to at least two vertices of B in G. Let G_{AB} be a bipartite subgraph of G on the $n_A + n_B$ vertices $A \bigcup B$ with $2n_A$ edges chosen so that each vertex of A is adjacent to exactly two vertices of B. An embedding of G in the plane provides an embedding of G_{AB} in the plane in which every face of the bipartite graph G_{AB} has an even number of boundary edges. We call a face of G_{AB} with four boundary edges a *4-face*. Let f_4 be the number of 4-faces of G_{AB}, and let f' be the number of faces with six or more boundary edges. Note that if G_{AB} is not connected, then some faces will have boundary edges from two or more components, yielding face boundaries with at least eight edges. It is immediate by Euler's formula that an embedding of a planar graph having p vertices, q edges, and c components must determine r faces where $p - q + r = 1 + c$. For G_{AB} we obtain $(n_A + n_B) - 2n_A + f_4 + f' = 1 + c$; hence

$$f_4 + f' > n_A - n_B. \tag{2}$$

Suppose some edge a_1, b_1, for $a_1 \in A$, $b_1 \in B$, is on the boundary of two 4-faces F_1, F_2 of G_{AB}. Then we must obtain faces as shown in Figure 3, where a_1, a_2, a_3 are each adjacent to both b_1 and b_2. Since a_1 has degree five in G, one of the 4-faces, say F_1, must have $k \geq 2$ vertices of the original embedding of G in its interior, including at least two vertices adjacent to a_i. G is maximal planar, so the induced subgraph F_1^* of G on the union of the k vertices interior to F_1 and $\{a_1, b_1, a_2, b_2\}$ triangulates the interior of F_1. Since any planar triangulation of the interior of a four cycle having a total of i vertices must have $3i - 7$ edges, F_1^* has $3(k+4) - 7 = 3k + 5$ edges.

[Figure 3]

Now a_1 has degree at least four in F_1^*, and at least two vertices of $\{b_1, a_2, b_2\}$ have degree at least three in F_1^* since the interior of F_1 is triangulated by F_1^*. The k vertices interior to F_1 each have the same degree in F_1^* as in G, which

falls between six and eleven, since they are vertices of G not in A or B. The sum of the vertex degrees of F_1^* thus is at least $6k+12$, which implies that F_1^* has at least $3k+6$ edges, a contradiction. Hence every edge of G_{AB} can be a boundary edge of at most one 4-face of G_{AB}.

Since each 4-face of G_{AB} is bounded by four of the $2n_A$ edges of G_{AB}, and each edge can occur on the boundary of at most one 4-face,

$$4f_4 \leq 2n_A.$$

Since each of the $2n_A$ edges of G_{AB} is on the boundary of two faces of G_{AB}, then also

$$4f_4 + 6f' \leq 4n_A$$

so that

$$6f_4 + 6f' \leq 5n_A. \tag{3}$$

From (2) and (3) we obtain $5n_A > 6(n_A - n_B)$, so that

$$6n_B > n_A, \tag{4}$$

a contradiction to (1). We thus conclude that not every vertex of A, i.e., not every vertex of degree five in G, can be adjacent to at least two vertices of B, i.e., to two vertices of degree at least twelve in G, which completes the theorem. □

4. The Batch Contraction Algorithm.

Our batch contraction planar graph algorithm operates in phases. Each phase processes a reduced planar graph G^*. If G^* has a vertex of degree four or less, the phase merely deletes such a vertex. If, on the other hand, G^* has minimum degree five, the phase determines an independent set I, with $|I| \geq n/12$, such that each vertex in I has degree five or six. The determination of such a set utilizes the following lemma.

Lemma 7. *If an n vertex planar graph G has minimum degree five, then*

$$|\{v \mid \textit{degree}(v) = 5\}| + |\{v \mid \textit{degree}(v) = 6\}| > n/2.$$

Proof. Let $n_i = |\{v \mid degree(v) = i\}|$. The summation of the vertex degrees yield twice the number of edges for any graph, and the number of edges in any n-vertex planar graph with $n \geq 3$ is at most $3n - 6$. Therefore

$$5n_5 + 6n_6 + 7(n - n_5 - n_6) \leq 6n - 12,$$

from which the lemma follows immediately. □

If the reduced planar graph G^* has minimum degree five we 6-color it in linear time and then determine the largest set of vertices of degree five or six that receive the same color. By Lemma 7 at least 1/12 of the vertices of G^* are in such a set. The importance of this process derives from our ability in one phase to contract the regions around all these vertices of degree five or six, forming a new reduced planar graph, such that a 5-coloring of the reduced graph will give a 5-coloring of G^*. We have previously noted that any vertex of degree five has two adjacent vertices that are themselves non-adjacent, and which we may therefore contract. We need a similar result for vertices of degree six.

Lemma 8. *Let v be any vertex of degree six in an embedded planar graph G. Then either (i) there are three pairwise non-adjacent vertices u_1, u_2, u_3 all adjacent to v, or (ii) there are four vertices u_1, u_2, w_1, w_2, all adjacent to v, such that u_1 and u_2 are non-adjacent, w_1 and w_2 are non-adjacent, and the vertices appear in the order u_1, u_2, w_1, w_2 clockwise around v in the embedding.*

Proof. Suppose case (i) does not hold, i.e., out of any three vertices adjacent to v, two are themselves adjacent. Let x_1, x_2, x_3, x_4, x_5, x_6 be the vertices adjacent to v, in clockwise order around v. One of the edges (x_1, x_3), (x_3, x_5), (x_5, x_1) must be present in G; without loss of generality suppose it is (x_1, x_3). See Figure 4. The vertices x_1, x_3, v form a cycle with x_2 on one side and x_4, x_5, x_6 on the other side; thus x_2 is adjacent neither to x_4 nor to x_6. Since case (i) does not hold (x_4, x_6) must be an edge of G. The vertices x_4, x_6, v form a cycle with x_5 on one side and x_1, x_2, x_3 on the other side; thus x_5 is not adjacent to x_1, and the vertices $u_1 = x_2$, $u_2 = x_4$, $w_1 = x_5$, $w_2 = x_1$ satisfy the requirements of case (ii). □

[Figure 4]

We now present our batch contraction planar graph 5-coloring algorithm.

Algorithm BC [Batch Contraction 5-Coloring].

Given an n vertex planar graph G in adjacency list form, this algorithm determines a 5-coloring of G. In the following description G refers to the original planar graph and G^* refers to a graph that initially has the same structure as G, but which is reduced by deletions and contractions until it is the null graph. At all times each vertex of G^* corresponds to an independent set of vertices of G.

Step 1. [Establish adjacency structure and degree lists for G^*.] Create an adjacency list data structure for the graph $G^* = G$. For each $0 \leq j \leq n-1$, form a list of all vertices of G^* of degree j.

Step 2. [Delete minimum degree vertex in G^* if degree is at most four.] While the j-degree lists for $0 \leq j \leq 4$ are not all vacuous, determine v^* to be the first vertex in the non-vacuous j-degree list of smallest j. Delete v^* from G^* and add v^* to the stack of vertices deleted. Update the adjacency list data structure and degree lists for G^* to reflect the deletion of v^*.

Step 3. [Determine batch of vertices to be deleted if G^* has minimum degree five.] If G^* is now null, go to Step 6. Otherwise determine a set S composed of at least 1/12 of the vertices of G^* where each vertex of S has degree five or six in G^* and all vertices of S are colored the same in the 6-coloring.

Step 4. [Embed G^* in the plane.] Determine an embedding of the planar graph G^* and a clockwise ordering of the vertices adjacent to v^* for each vertex v^* of G^*.

Step 5. [Batch deletion and contraction.] For each vertex v^* in S of degree five, determine two non-adjacent vertices u_1, u_2 both adjacent to v^*. For v^* in S of degree six, determine a set of adjacent vertices satisfying Lemma 8. Now update the adjacency structure and degree lists for G^* to correspond to all of the following operations. Successively delete each vertex v^* of S^* from G^* and add v^* to the stack of vertices deleted. For each v^* in S which had degree five, contract u_1, u_2 into a new vertex. If v^* had degree six and satisfied (i) of Lemma 8, contract u_1, u_2, u_3 into

a new vertex. If v^* satisfied (ii) of Lemma 8, contract u_1, u_2 into a new vertex and w_1, w_2 into another new vertex. See Figure 5. Note that since the vertices in S are pairwise non-adjacent, these contractions do not interfere with each other and can be carried out in one process. The new reduced graph G^* will then have at most 5/6 the number of vertices in G^* prior to the batch deletion and contraction process. Return to Step 2.

Step 6. [Color vertices of G.] While the stack of vertices deleted from G^* is not empty, remove v^* from the stack. Traverse the independent set $\{v_1, v_2, \ldots, v_k\}$ of vertices of G corresponding to v^* and color all these vertices with the smallest positive integer value [which must be between one and five] that has not previously been assigned as the color of any vertex of G adjacent to any vertex of the set $\{v_1, v_2, \ldots, v_k\}$.

[Figure 5]

Utilizing Lemmas 7 and 8, the correctness of Algorithm BC follows in a straightforward manner annalogous to the proof of Theorem 5 for Algorithm SC. We now consider the running time of Algorithm BC.

Theorem 9. *Algorithm BC can be implemented to run in* $O(n)$ *time.*

Proof. Steps 1 and 6 are each executed only once and each can readily be implemented to run in $O(n)$ time [see Lemma 2 and Theorem 5].

Steps 2, 3, 4, and 5 are executed repeatedly, but each time with a graph having at most 5/6 the number of vertices in the previous graph. Let n^* be the number of vertices of G^* in a given execution of Steps 2 – 5. The deletions of Step 2 can be implemented in $O(n^*)$ time. Using Algorithm 6-COLOR and noting the result of Lemma 7, Step 3 can be implemented in $O(n^*)$ time. Step 4 can be implemented in $O(n^*)$ time by procedures available in the literature [HT 74]. Thus our concern is to show that Step 5 can be implemented in linear time.

We carry out Step 5 as follows. We first construct the set of triples $T = \{(u, w, v) \mid v \in S, u \text{ and } w \text{ are adjacent to } v, \text{ and } u \neq w\}$. (Note that $|T| = O(n)$.) Using a two-pass radix sort [K 73], we sort T lexicographically on its first two components. We then compare the sorted list of T with a lexicographically

ordered list of the edges in G^*. This comparison allows us to associate with each triple (u, w, v) a Boolean value $B(u, w, v)$ such that $B(u, w, v)$ is true if and only if (u, w) is an edge of G^*. Another radix sort of T, this time on the third component, allows us to associate with each vertex v the set of triples $(u, w, v) \in T$ along with the values $B(u, w, v)$. Given this adjacency information, finding vertices satisfying the contraction conditions requires constant time per vertex of S. As in Algorithm SC, we use circular lists to represent the independent sets corresponding to each vertex of G^*. Thus each contraction step takes constant time. Each vertex v^* of the graph G^* before contraction corresponds to some (possibly new) vertex y^* of the new graph G^* resulting from the contraction. Given these vertex identifications, the desired adjacency structure and degree lists for the new reduced graph G^* can be constructed in $O(n^*)$ time. Note that a radix sort can be used to order the adjacency lists of each new vertex y^* of G^* to eliminate any duplicate edges created by the contractions. The radix sorts used to process the triples of T and to eliminate duplicate edges in the reduced graph G^* require only $O(n^*)$ time [K 73].

Hence Steps 2, 3, 4, and 5 can be implemented in time proportional to the number of vertices in G^* for each pass through these steps. Since each pass involves a graph having at most 5/6 as many vertices as the graph in the previous pass, the total time for Steps 2 - 5 is bounded by $cn(1 + (5/6) + (5/6)^2 + \cdots) = (cn)/(1 - (5/6)) = 6cn$ for some constant c. Hence Algorithm BC can be implemented to run in $O(n)$ time. □

5. Remarks.

There are several interesting features of our two linear-time five-coloring algorithms that we would like to summarize here.

(i) Our algorithms each exhibit a strong relation between an inductive proof of a theorem and the design of a recursive algorithm

(ii) The algorithms show that different inductive proofs of the same theorem may suggest different recursive algorithms with different time bounds.

(iii) The algorithms succinctly contrast the sequential processing and batch processing approaches to achieving linear time through problem reductions.

(iv) Algorithm SC is purely combinatorial requiring no knowledge of any embedding of the planar graph, whereas Algorithm BC requires both combinatorial and topological information.

(v) The essential importance of data structure design is apparent in both algorithms. In particular, note that a trivial question related to one data structure [e.g. determination of a particular non-adjacency given the adjacency matrix] can be quite complex to handle with satisfactory efficiency in another data structure [e.g. determination of non-adjacency given only the adjacency list structure of the graph].

The recent proof by Appel and Haken [AH 76] that every planar graph can be 4-colored resolved a famous problem of long standing. Their proof is exceedingly tedious, but it leads to an $O(n^2)$-time algorithm for 4-coloring a planar graph. The problem of finding a linear-time 4-coloring algorithm remains open.

References

[AH 76] K. Appel and W. Haken, "Every planar map is four colorable, Part I: discharging," *Illinois J. of Math.* 21 (1977), 429–490 [see also Part II, ibid. 491–567].

[BC 71] M. Behzad and G. Chartrand, *Introduction to the Theory of Graphs*, Allyn and Bacon, Boston, 1971.

[B 62] C. Berge, *The Theory of Graphs and Its Applications*, John Wiley, New York, 1962.

[B 79] B. Bollobas, *Graph Theory*, Springer-Verlag, New York, 1979.

[BM 76] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, American Elsevier, New York, 1976.

[H 68] F. Harary, *Graph Theory*, Addison-Wesley, Reading, Mass., 1969.

[H 90] P. J. Heawood, "Map color theorems," *Quart. J. Math.* 24 (1890), 332–338.

[HT 74] J. E. Hopcroft and R. E. Tarjan, "Efficient planarity testing," *Journal ACM* (1974), 549–568.

[K 73] D. E. Knuth, *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison-Wesley, Reading, Mass., 1973.

[LM 78] R. J. Lipton and R. E. Miller, "A batching method for coloring planar graphs," *Information Processing Letters* 7 (1978), 185–186.

[LT 79] R. J. Lipton and R. E. Tarjan, "A separator theorem for planar graphs," *SIAM J. App. Math.* 36 (1979), 177–189.

[MMI 72] D. W. Matula, G. Marble, and J. D. Isaacson, "Graph coloring algorithms," in *Graph Theory and Computing*, ed. R. C. Read, Academic Press, New York, 1972, 109–122.

[S 79] Y. Shiloach, "Union-member algorithms for non-disjoint sets," Technical Report STAN-CS-728, Computer Science Department, Stanford University (1979), submitted to *Journal ACM*.

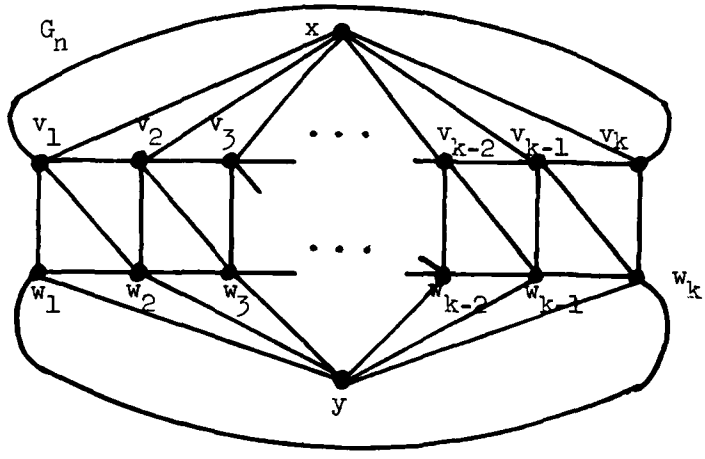


Figure 1. A planar graph on $n = 2k+2$ vertices where every vertex of minimum degree five has the sum of the degrees on its adjacent vertices equal to $k+20$ for any $k \geq 6$.

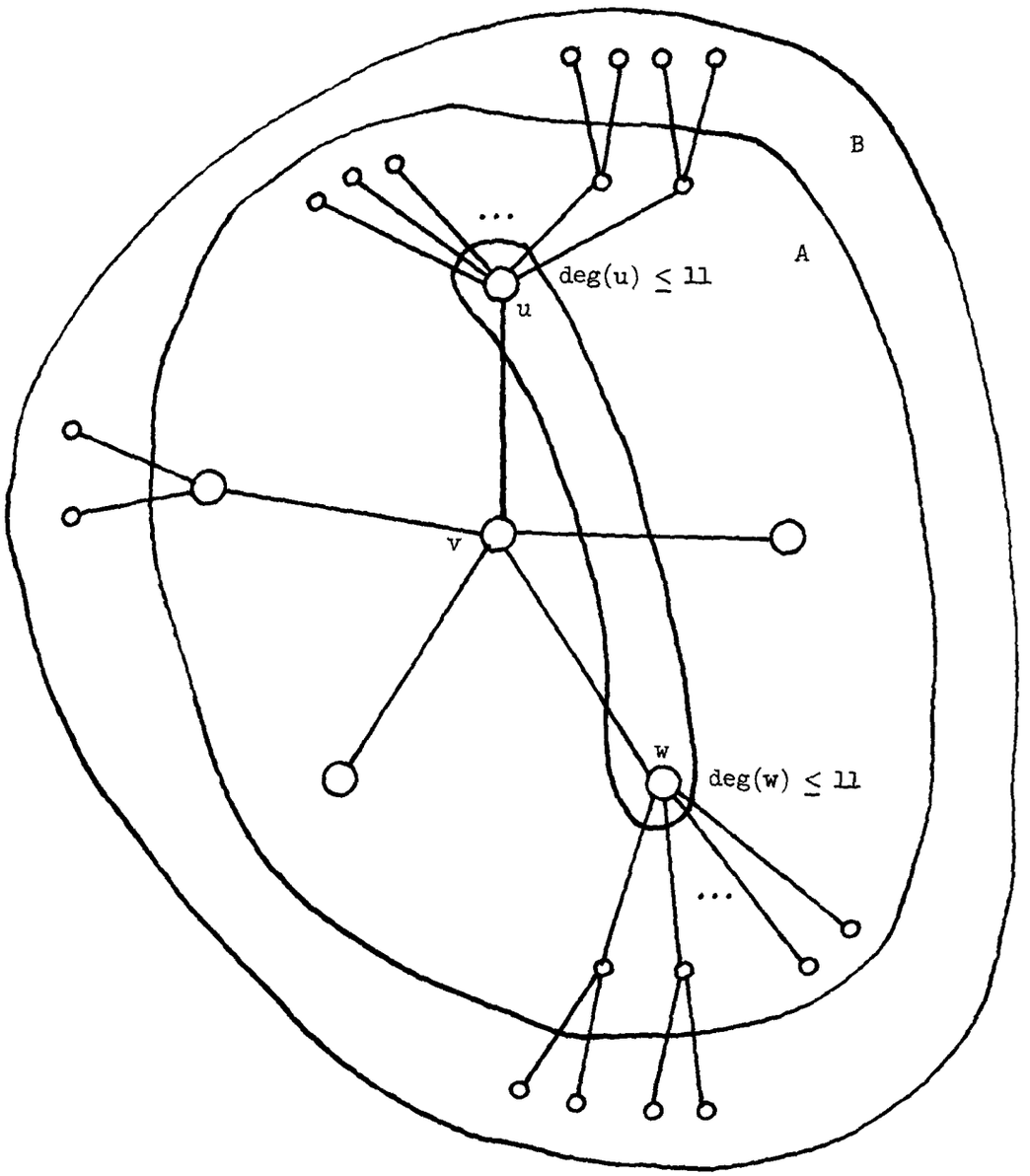


Figure 2. Deletion and contraction step. Vertices in region A will have their adjacency lists updated. Vertices shown in B have degree five and need to be placed in appropriate $5^{[11]}$- or 5-degree lists.

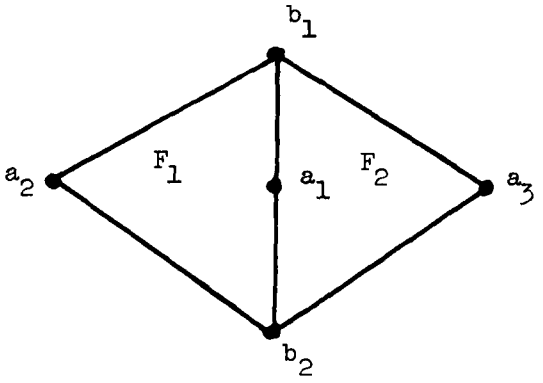


Figure 3

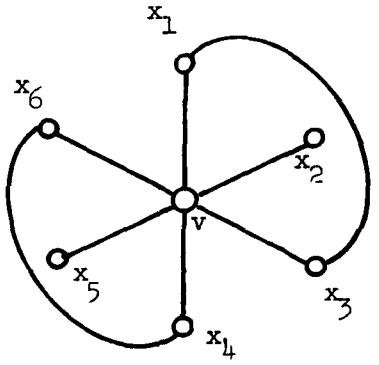


Figure 4. Degree six vertex for which Case (i) does not hold.

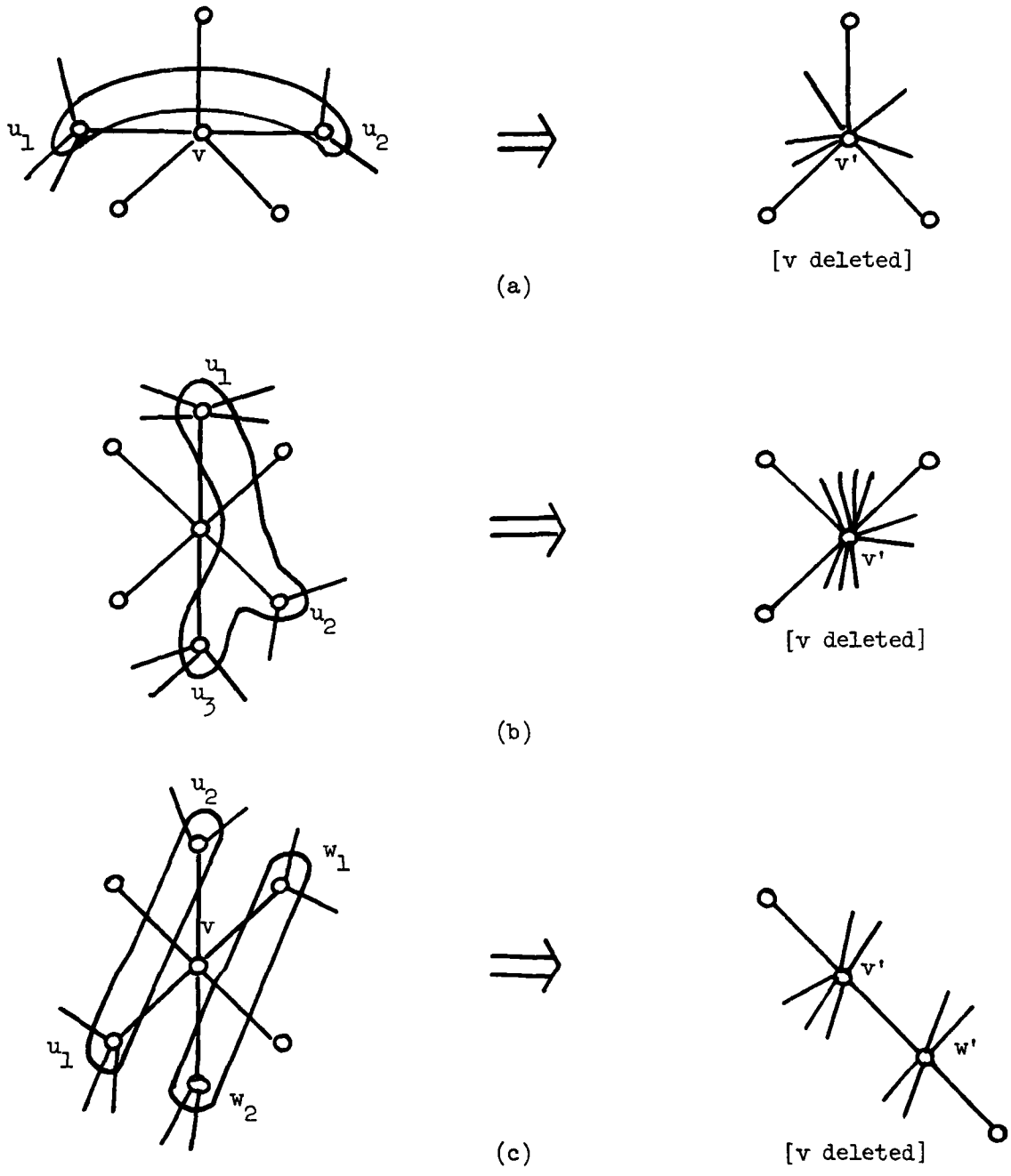


Figure 5. Deletion and contraction near vertices in S .

(a) Degree five vertex.

(b) Degree six vertex, Case (i).

(c) Degree six vertex, Case (ii).


DAT
ILM